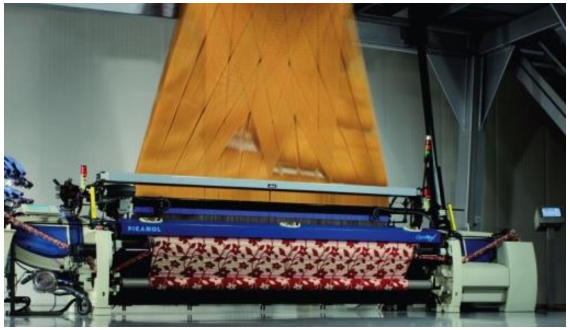
USING PAPYRUS IN A DESIGN SPACE EXPLORATION TOOLCHAIN

CURRENT DEVELOPMENTS AT FLANDERS MAKE

Who is Flanders Make?

A Flemish research institute whose mission is
to strengthen the **long-term international competitiveness** of the Flemish manufacturing industry by carrying out **excellent, industry-driven, pre-competitive research** in the domains of

- **Mechatronics**
- **Product development methods**
- **Advanced manufacturing technologies**



Aiming at product & process innovation
for the **vehicles, machines** and **factories of the future**

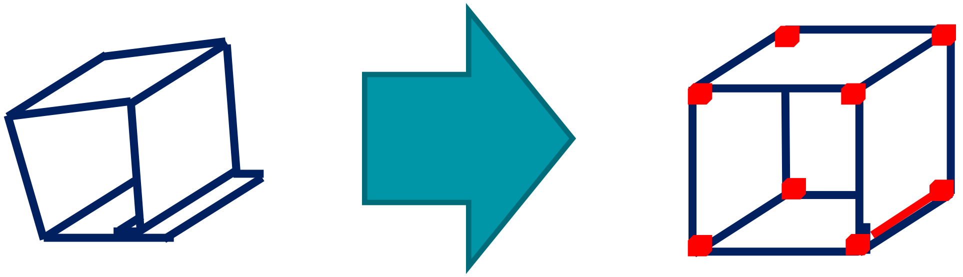
Our partner network

Outline

- Use case
  - Introduction
  - Design space model
  - Toolchain
- Discussion
  - Priorities
  - Tool usability
  - OCL framework
  - Instance creation and visualization

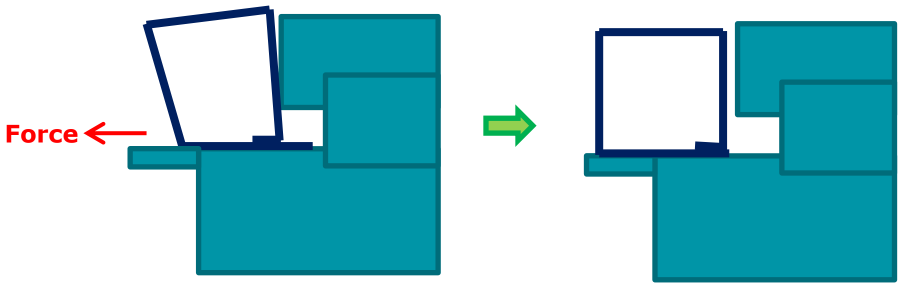
Conceptual design of a robotic assembly cell

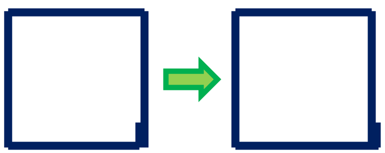
- Product to be manufactured
  - Casing for valve in ventilation system
- Requirements
  - Produce 40 cases per hour
  - Fully automated, no operator involved
  - Use existing machine repository
  - Design cell with minimal cost

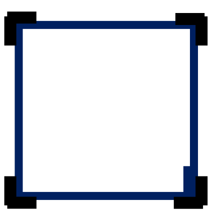


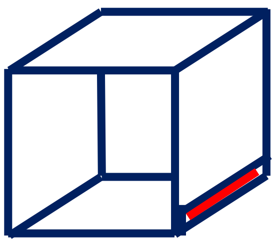
Conceptual design of a robotic assembly cell

- Process steps
  - Extract from bending machine
  - Correct folding
  - Join corners
  - Join seam









Conceptual design of a robotic assembly cell

- What is the best way of producing this product?
  - What machines do we use per process step?
    - Different ways ("working principles") to perform a process step
  - How are these machines assigned to the process steps?
    - Machines can be shared between steps (speed vs cost)

Correct Folding → Join Corners → Join Seam

Static machine?

Robot with end- effector?

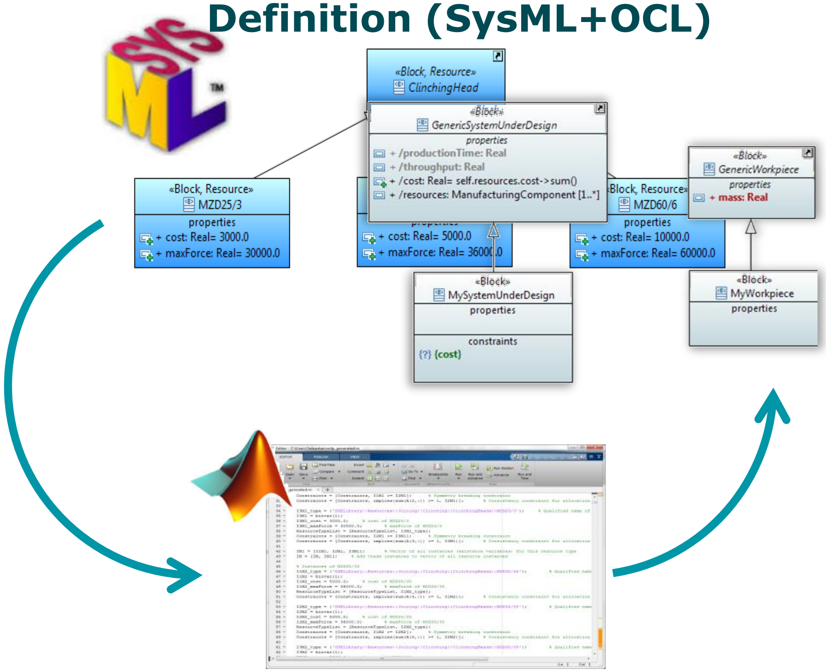
Separate machine?

Conceptual design of a robotic assembly cell

- Approach: Computational design synthesis
  - Represent design problem in a formal model (SysML+OCL)
  - Design repository to store knowledge for computational synthesis
  - Automated transformation to a Mixed-Integer Linear Program (MILP)
  - Represent solution in SysML

→ Explore more of the design space at a lower cost

Design Repository + Problem Definition (SysML+OCL)

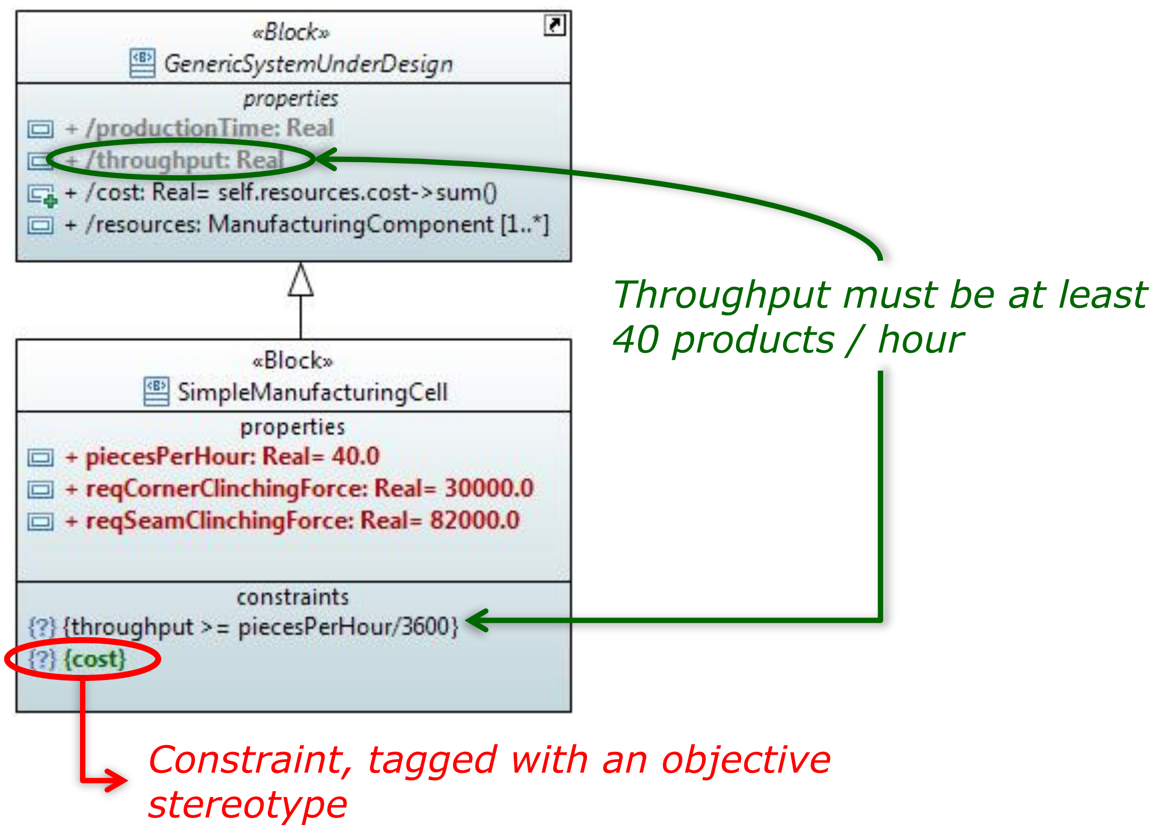


Analysis & Evaluation (MILP)

Design space model

Problem definition

- System-Under-Design, Objective & Requirements



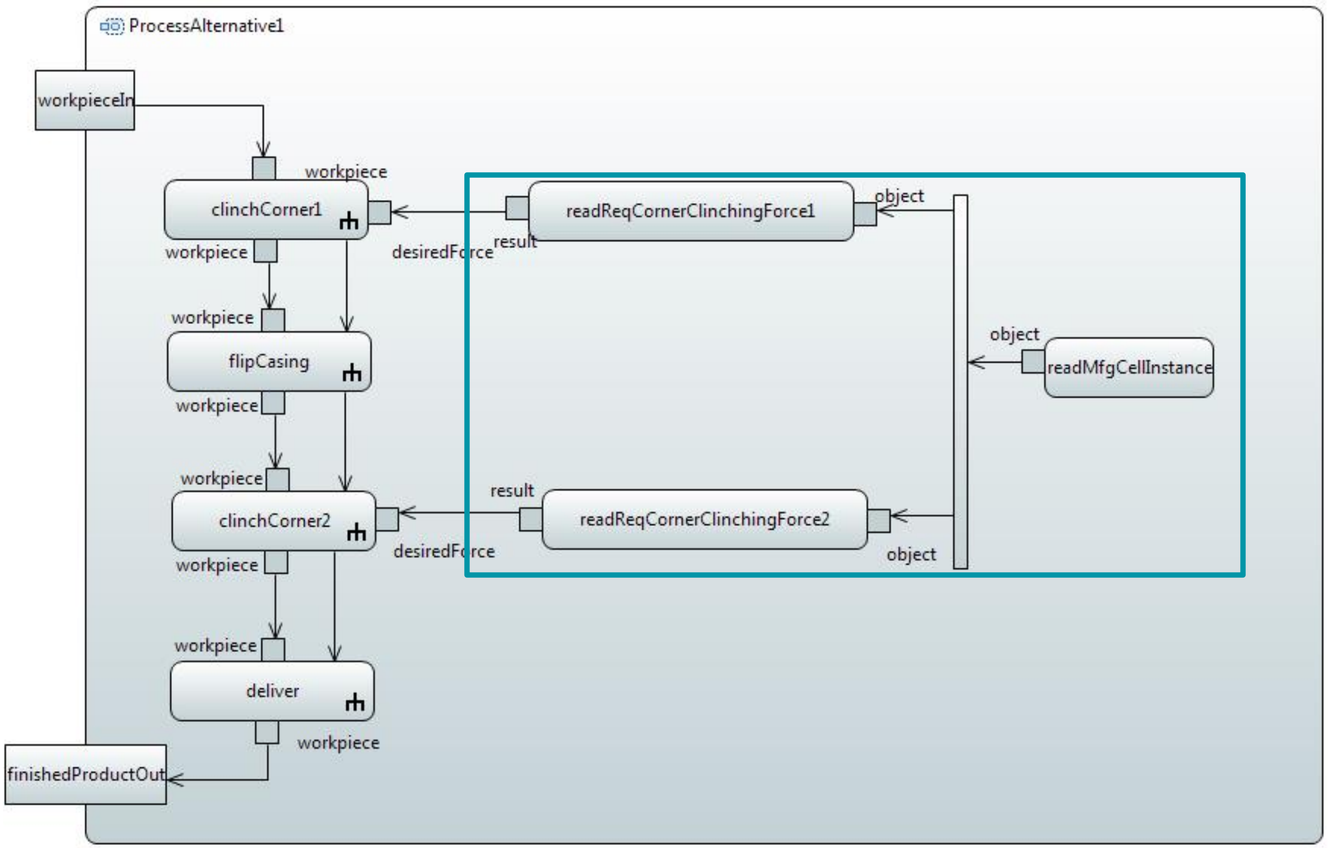
SysML 1.1

Design space model

Problem definition

Suggestions for more intuitive representations?
- Textual?

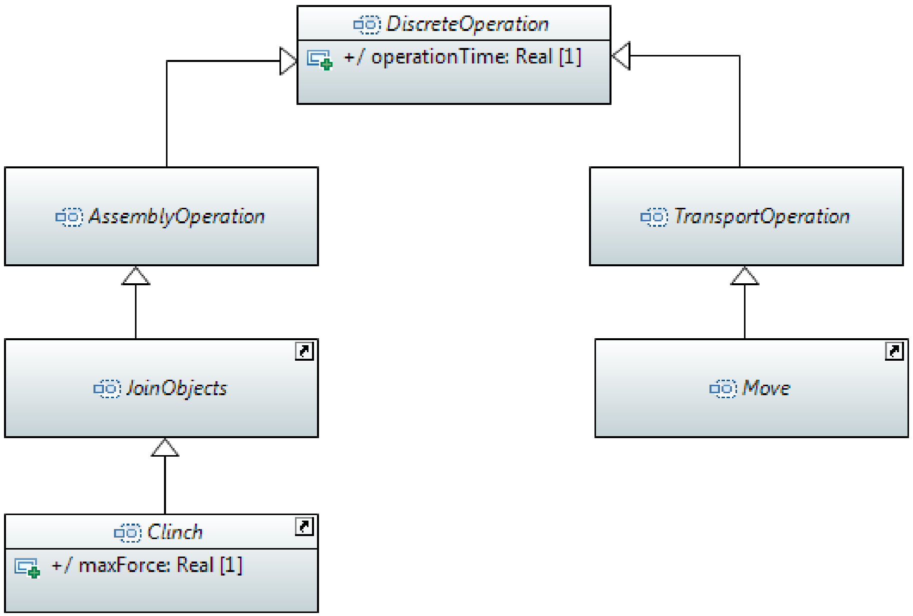
- Functional specification
  - Simplified sequential process shown

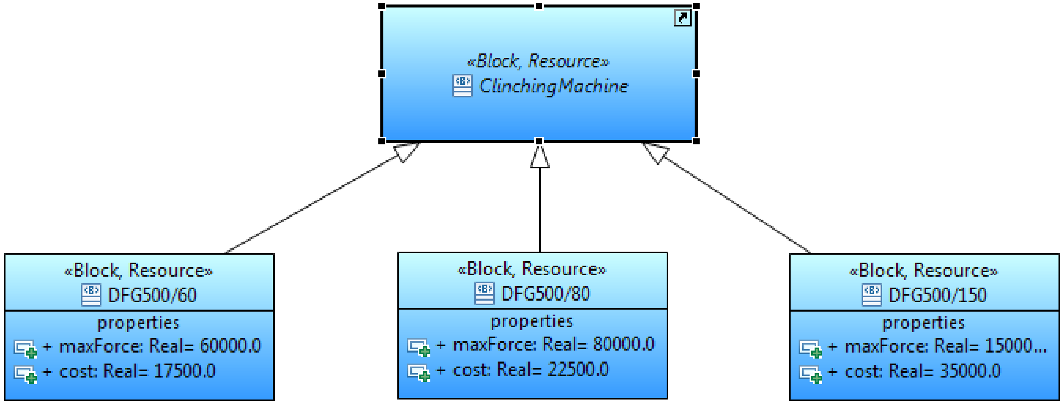


Design space model

Design repository

- Activities
  - Activity hierarchy
  - Assumes property and parameter equivalence
    - Inheritance
    - Comparison through OCL constraints
    - Visualization in class diagram
- Resources
  - Resource hierarchy
  - Non-abstract leaves redefine all inherited properties and specify their values

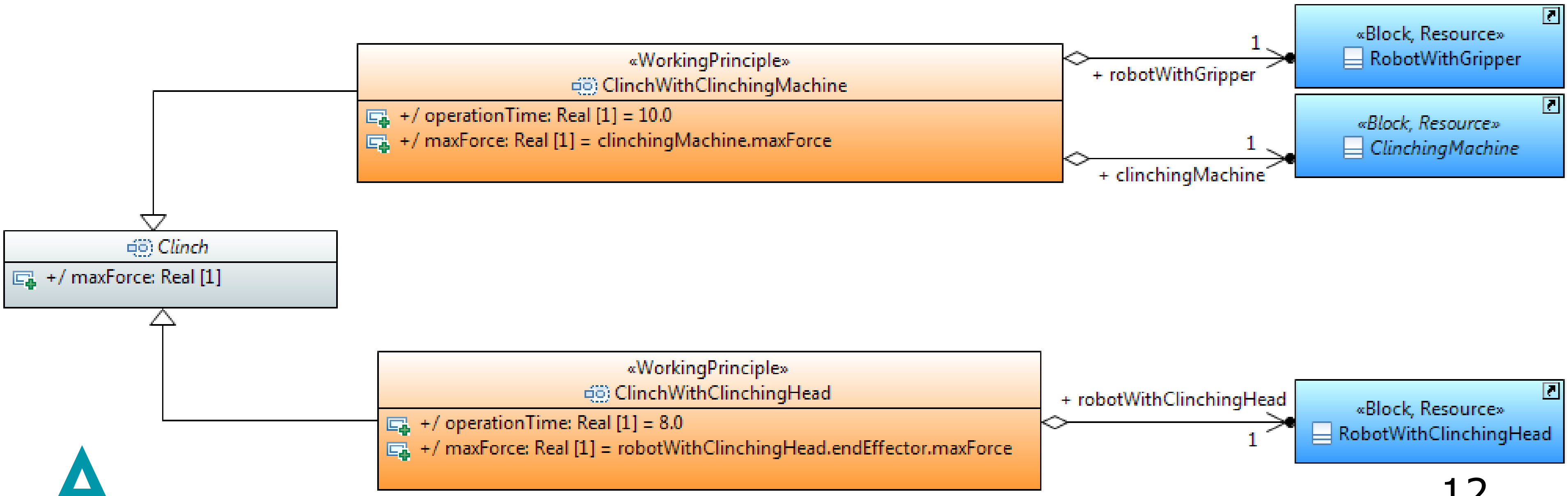




Design space model

Design repository

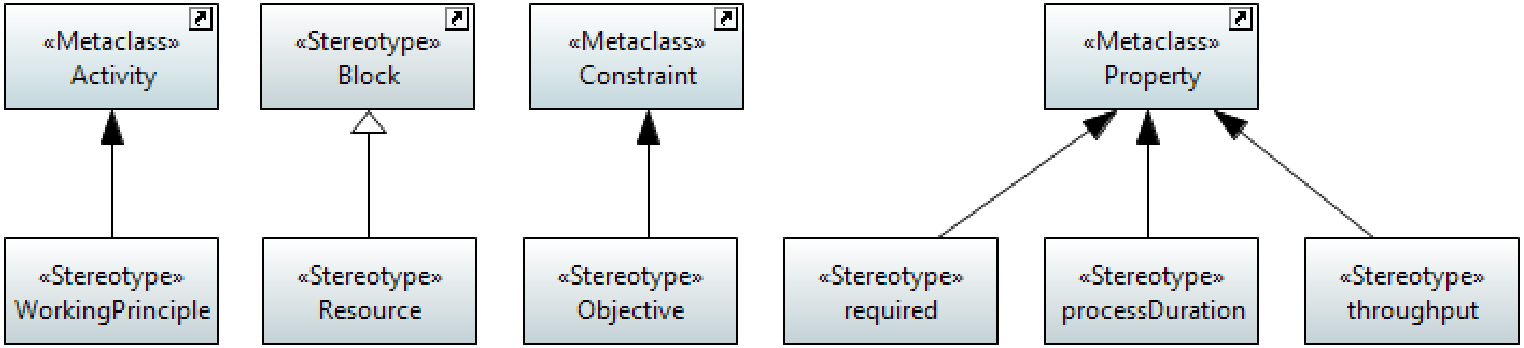
- Working Principles
  - Non-abstract leaves in the activity hierarchy
  - Links to the resources needed to execute the activity
  - Redefines properties and fills in their value, often based on resources used



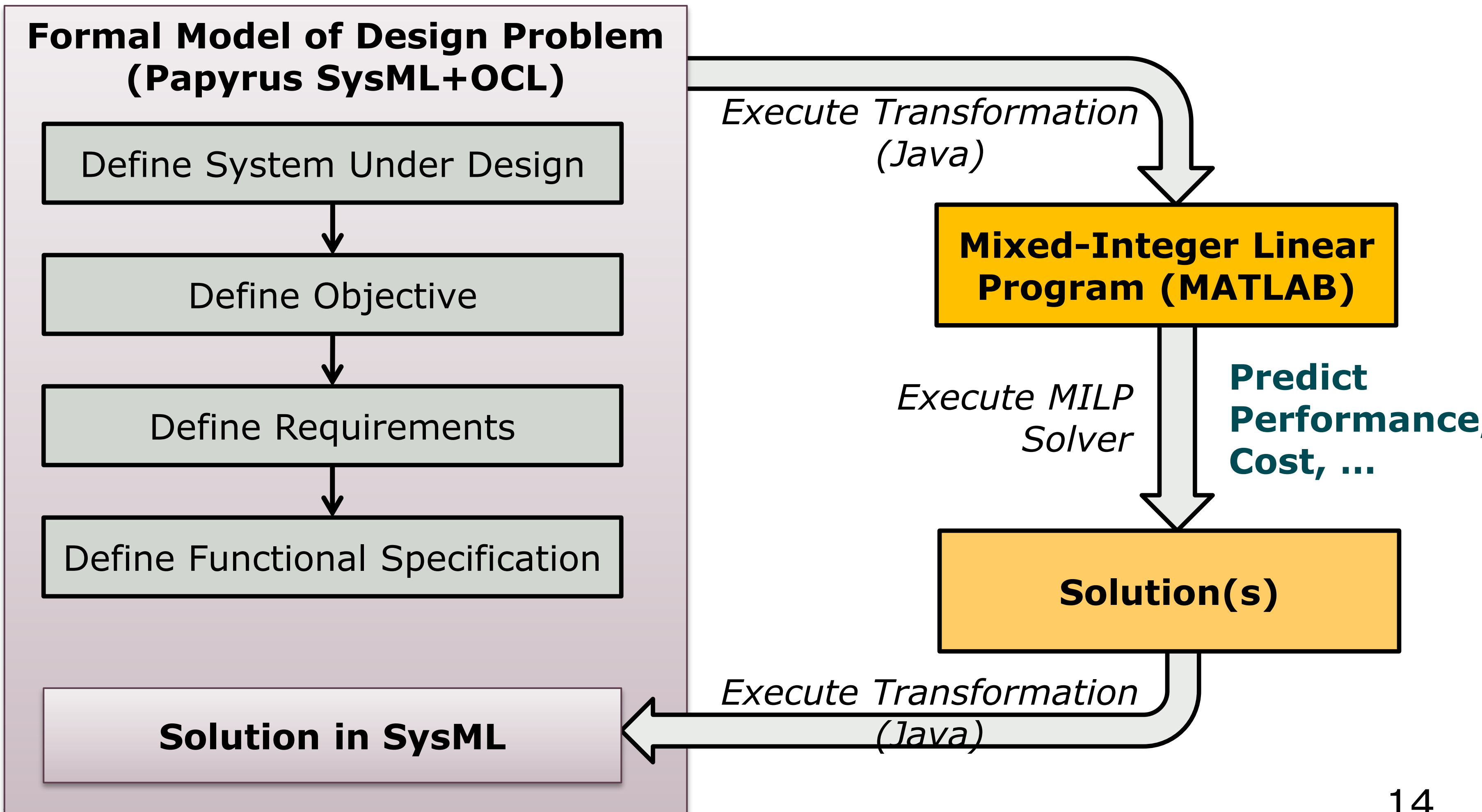
Design space model

Specialized profile

- Extensive use of existing concepts
  - Inheritance, redefinition (not present in ecore), derived union,…
- Limited set of problem-specific concepts needed
  - Some only needed because of validation

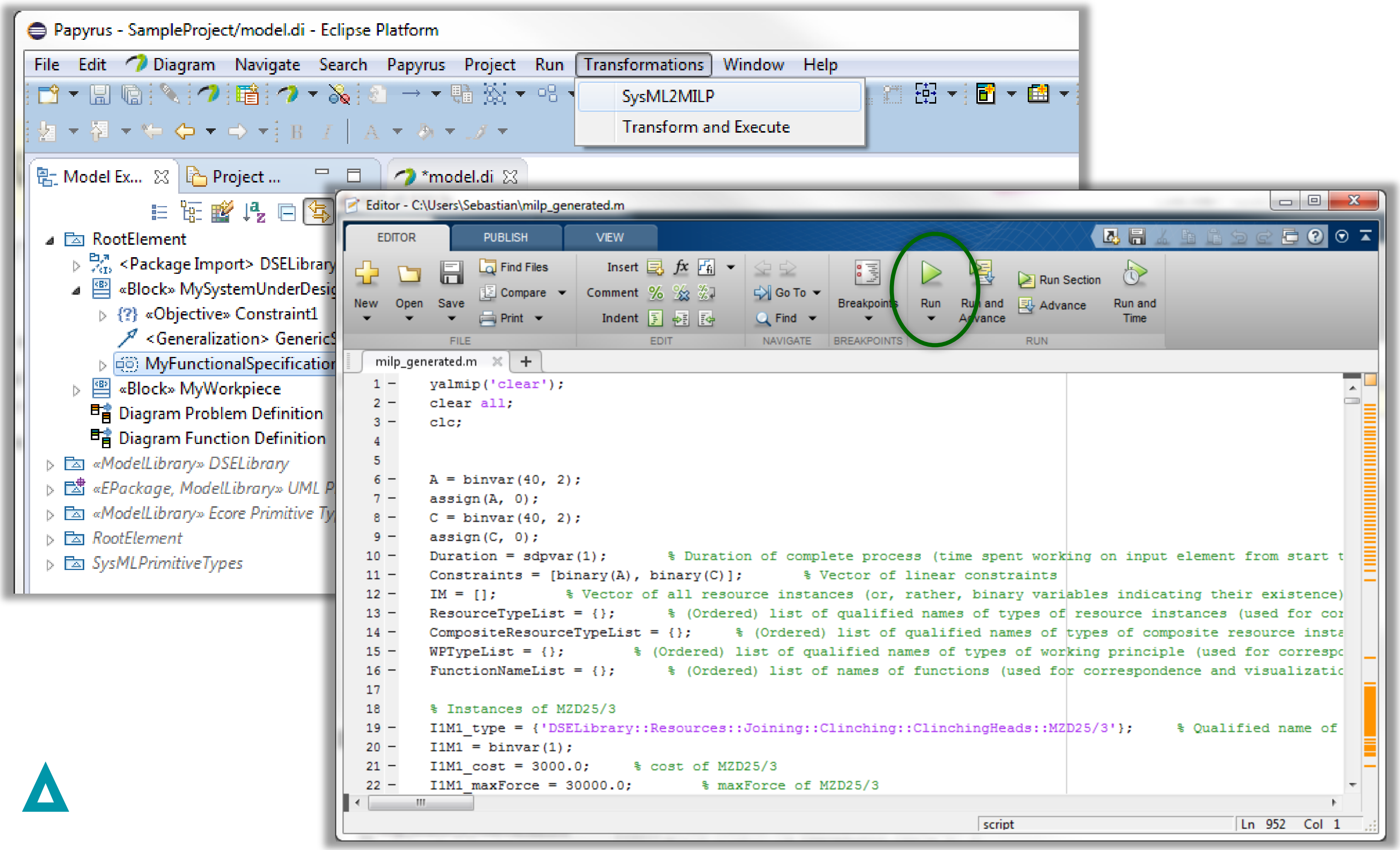


Toolchain

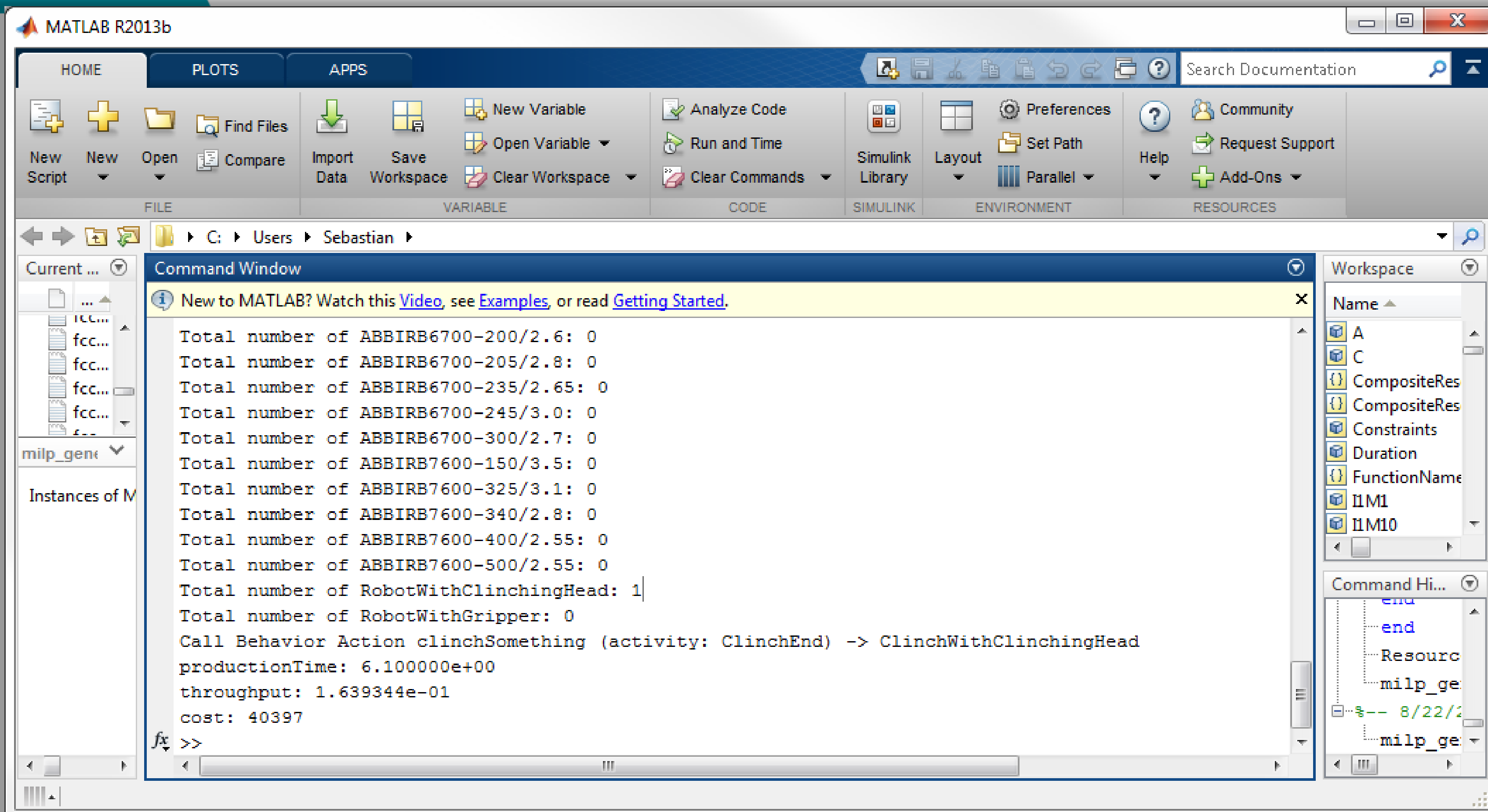


Formal Model of Design Problem
(Papyrus SysML+OCL)
Define System Under Design
Define Objective
Define Requirements
Define Functional Specification
Solution in SysML
Execute Transformation
(Java)
Mixed-Integer Linear Program (MATLAB)
Execute MILP Solver
Predict Performance, Cost, …
Solution(s)
Execute Transformation
(Java)

Automated Transformation to MILP & Execution of Solver



Generation of Solution Instance(s)



Excel Output

Resources

Function

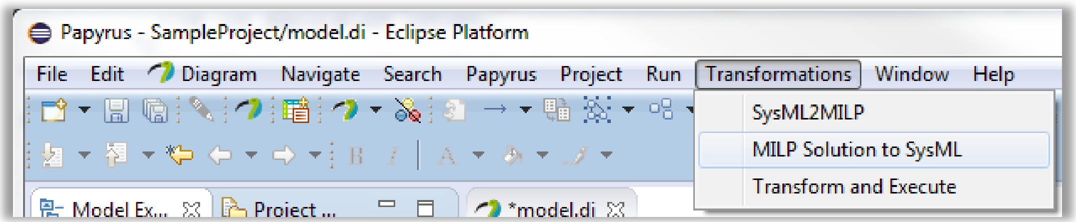
Working Principle

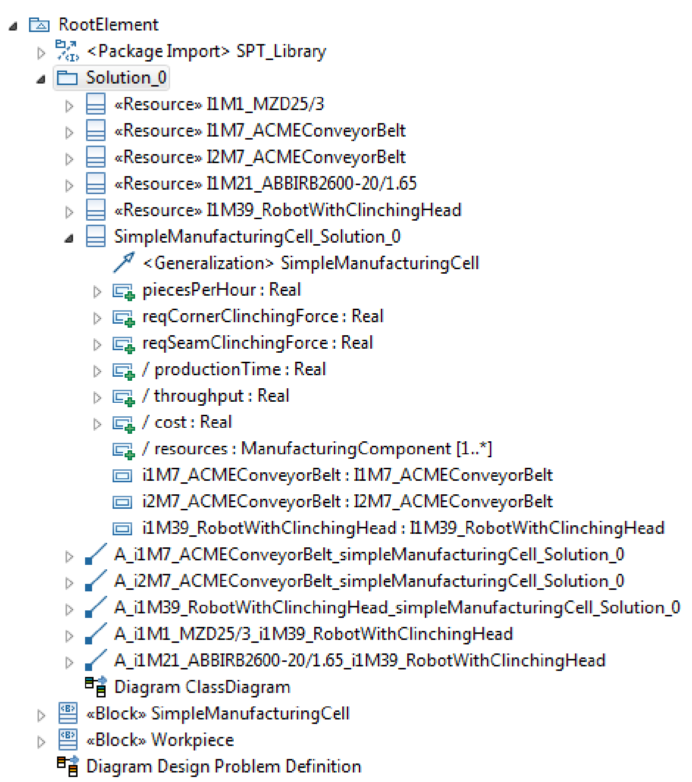
	A	B	C	
1		clinchCorner1 (ClinchWithClinchingHead)	clinchCorner2 (ClinchWithClinchingHead)	fl
2	MZD25/3	1	1	
3	ACMEConveyorBelt	0	0	
4	ABBIRB2600-20/1.65	1	1	
5	RobotWithClinchingHead	1	1	

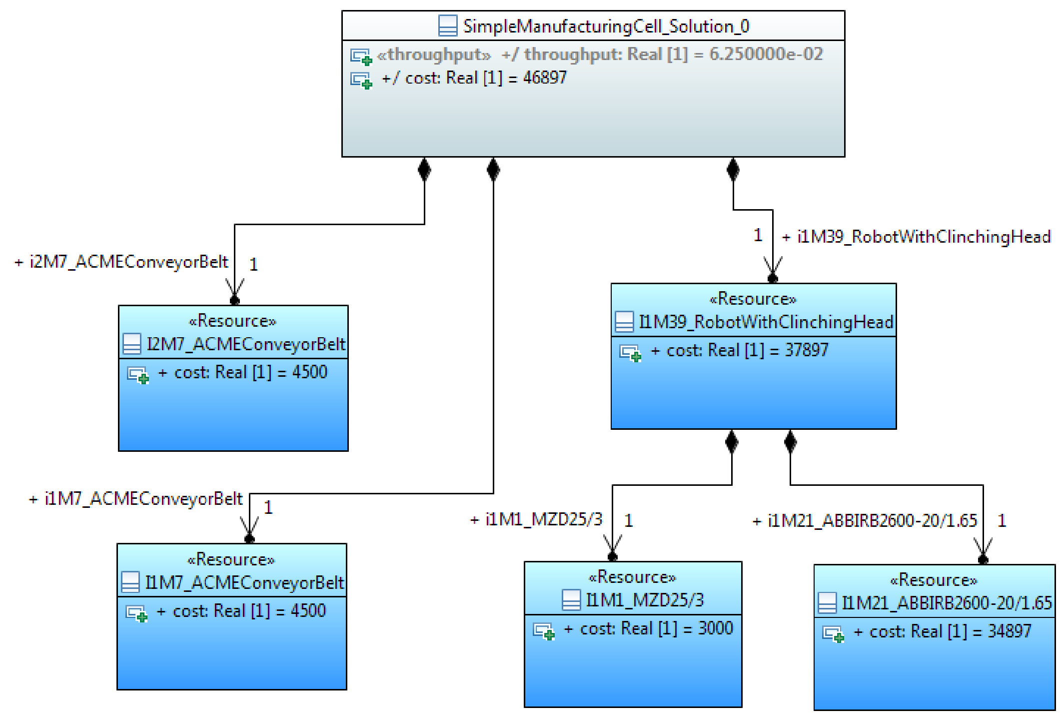
"1" if function allocated to resource

	A	B
1		RobotWithClinchingHead
2	MZD25/3	1
3	ABBIRB2600-20/1.65	1
4	RobotWithClinchingHead	1

Back Transformation







Outline

- Use case
  - Introduction
  - Design space model
  - Toolchain
- Discussion
  - Priorities
  - Tool usability
  - OCL framework
  - Instance creation and visualization

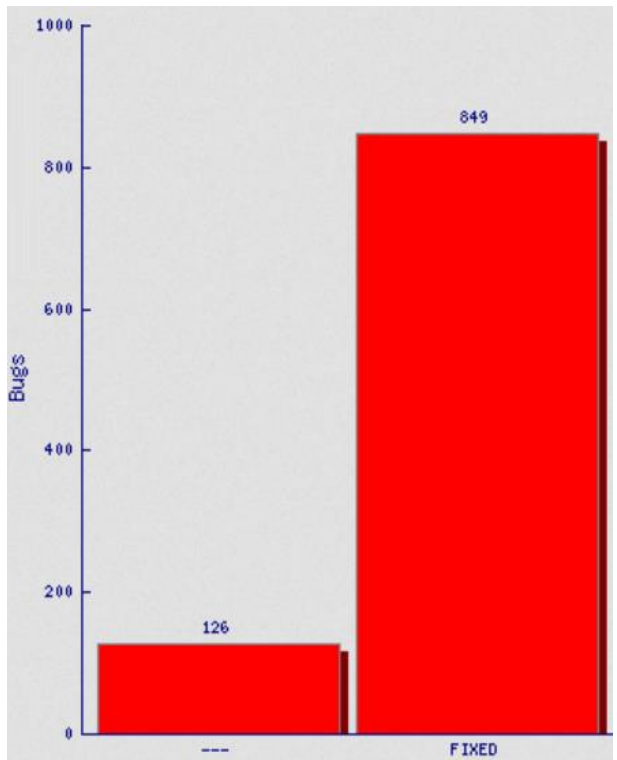
All of these points have been added to the bugzilla

Priorities

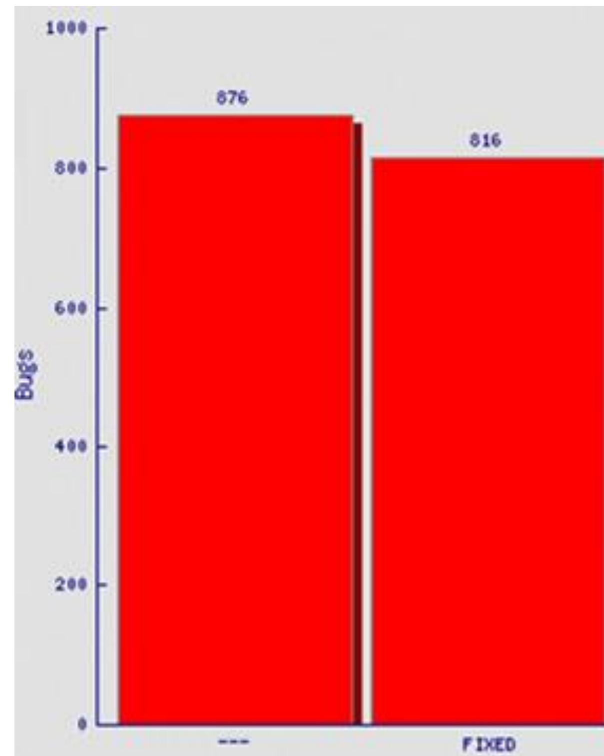
0 Governance
- Clear priorities
- Bugzilla

1. Usability
   - Simplification
   - Documentation
   - User-friendliness
   - Customization
2. Robustness
3. Communication
4. Features

2010-2011



01/01/2015-now



Priorities

0 Governance

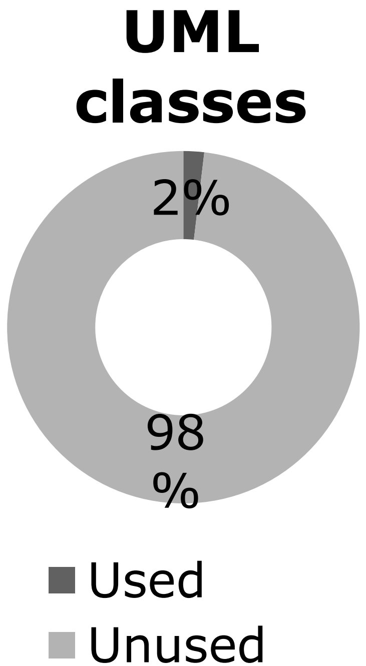
1. **Usability**
   1. **Simplification**
   2. **Documentation**
   3. **User-friendliness**
   4. **Customization**

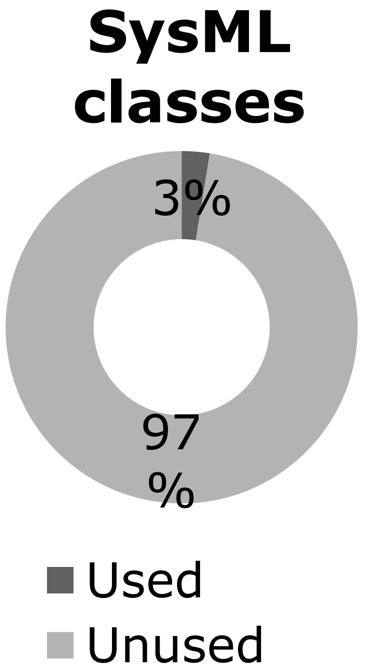
You can implement all the features we are asking for, but if the tool is too difficult too use, we still won't be able to get any of our member companies interested

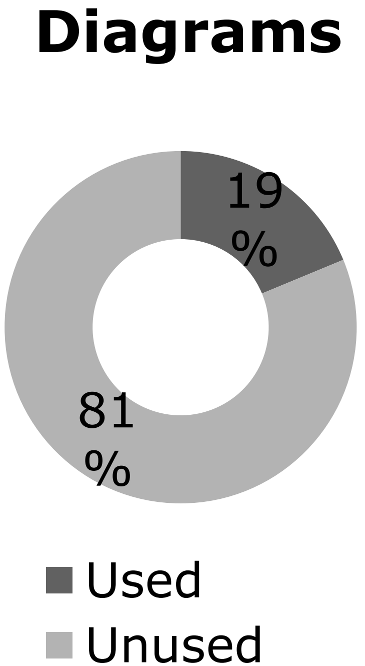
2. Robustness
3. Communication
4. Features

Tool usability

- Shield the user from the complexities of SysML
  - Typical DSL only needs a limited set of UML and SysML concepts

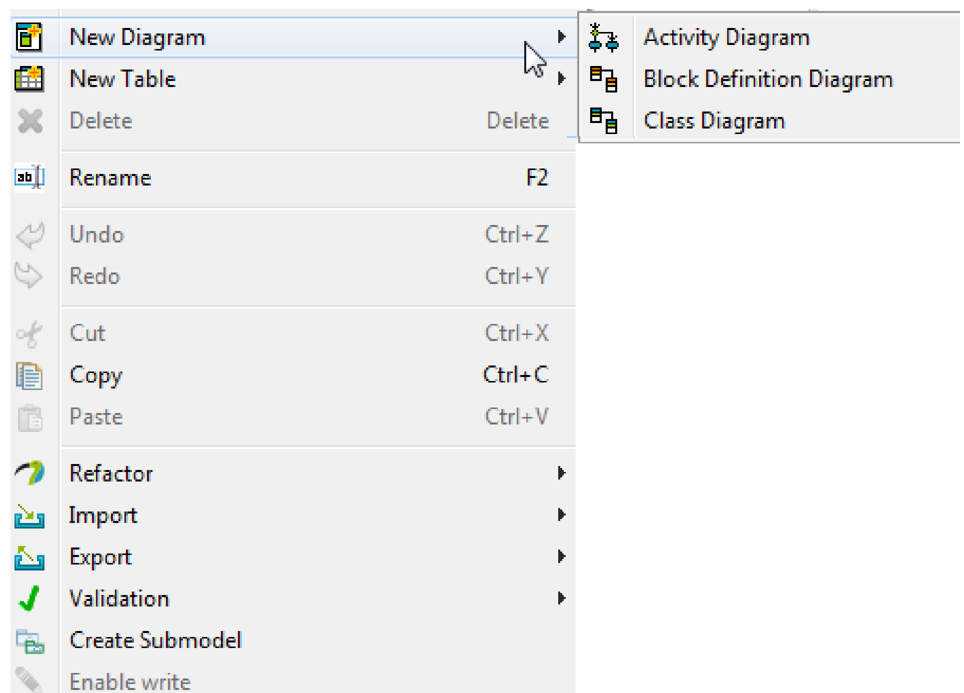






Tool usability

- Filtering of the content the user can use
  - Limit the number of diagrams/tables
  - Manipulate the new child menus
    - Single new child menu, containing just the elements we need
  - Simplification of the palette



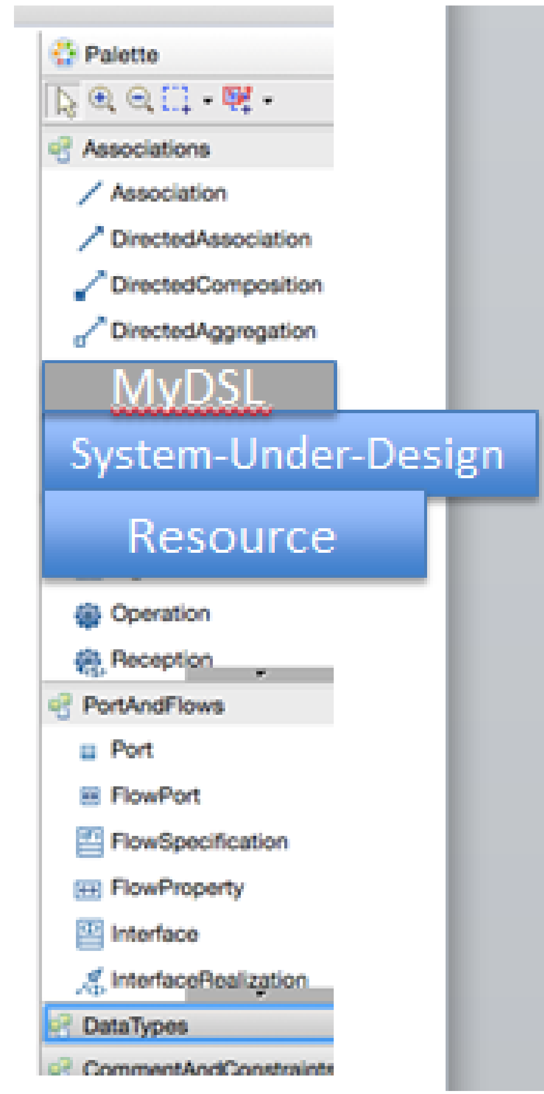
+ documentation of these features

Tool usability

- Add elements to the palette and menus with more complex functionality
  - Pre-stereotyped elements
  - Elements inheriting from a library element

  - Synchronization between palette and new child menu customizations?
    - Currently requires double work

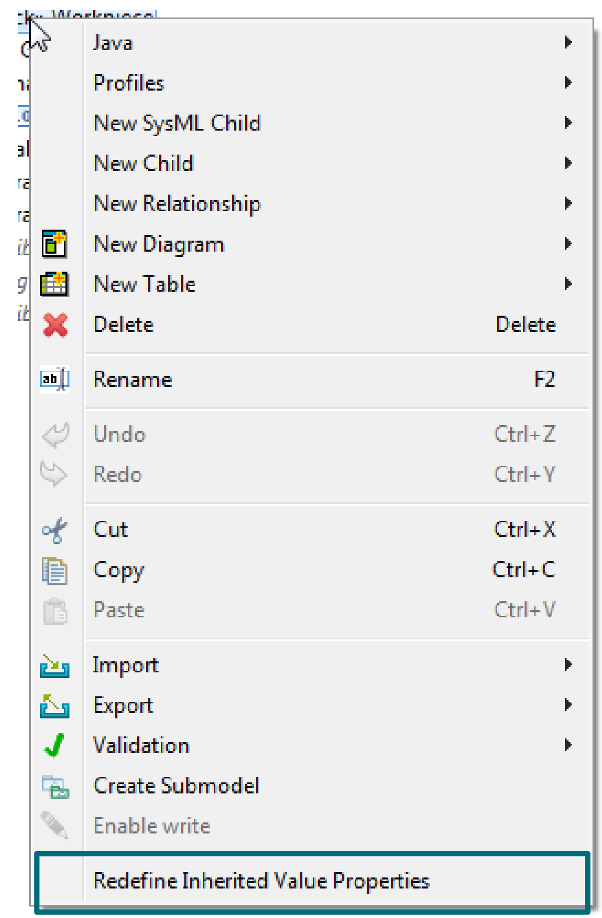
+ documentation of these features



Tool usability

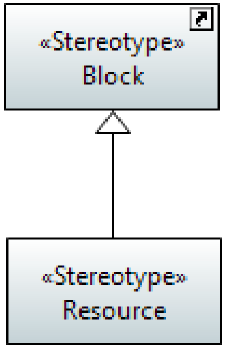
- Add elements to the palette and menus with more complex functionality
  - Pre-stereotyped elements
  - Elements inheriting from a library element
  - Complex functions to perform common tasks
    - Redefinition of all inherited properties

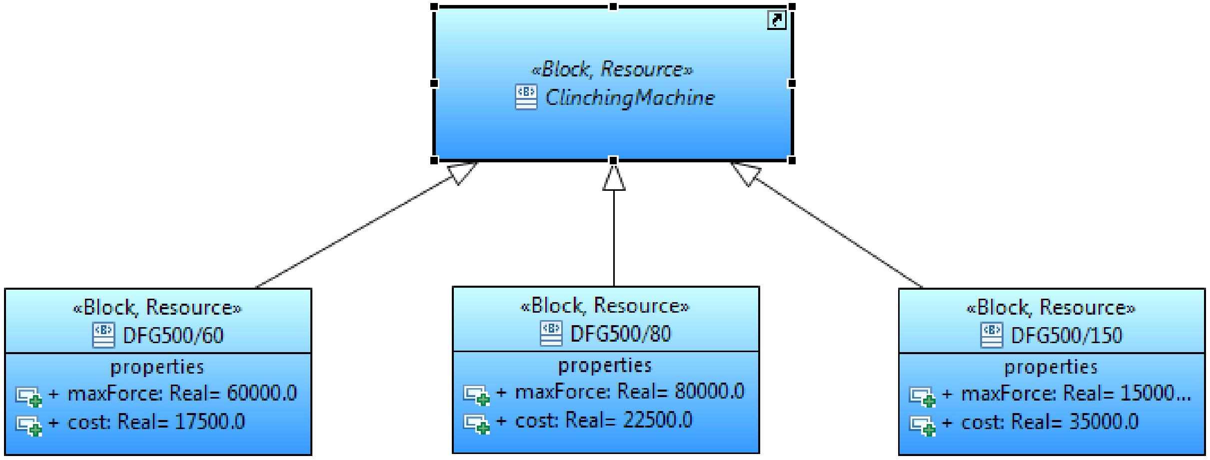
+ documentation of these features



Tool usability

- Additional validation rules



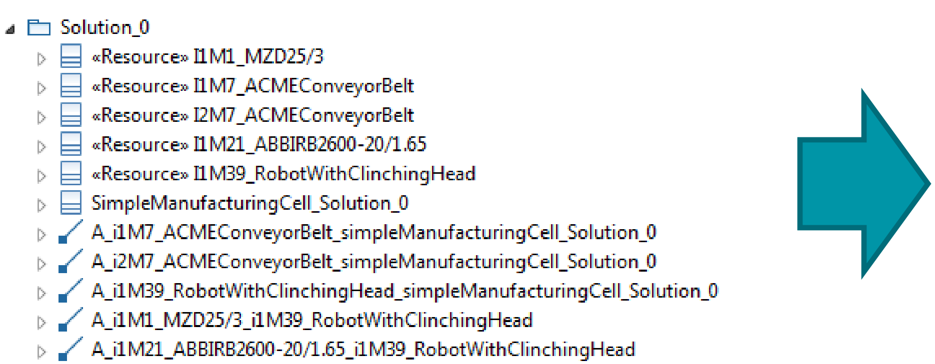


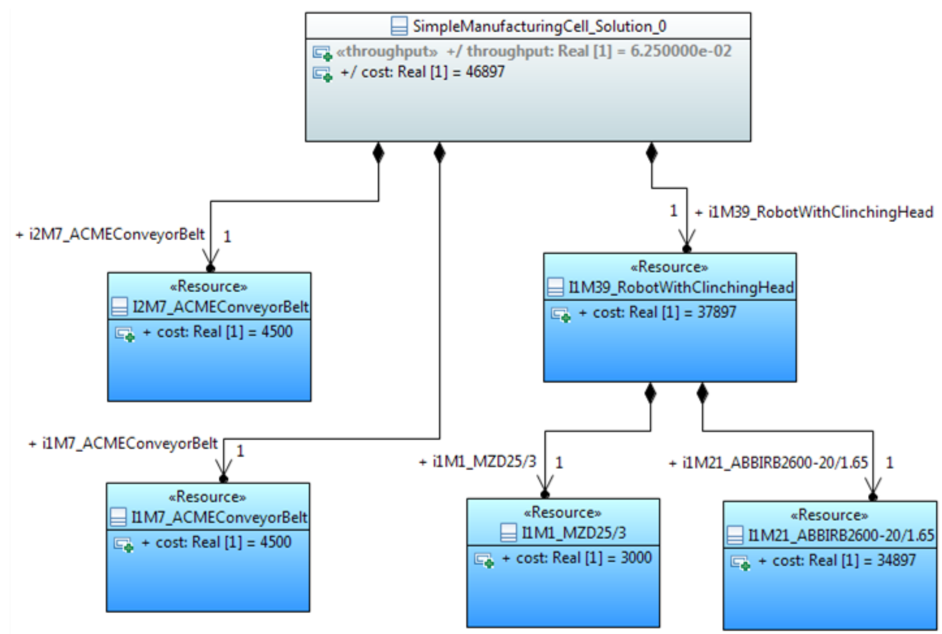
Tool usability

- Intelligent lay-outing of diagrams
  - Automatic lay-outing
  - Semantic lay-outing

+ documentation of these features

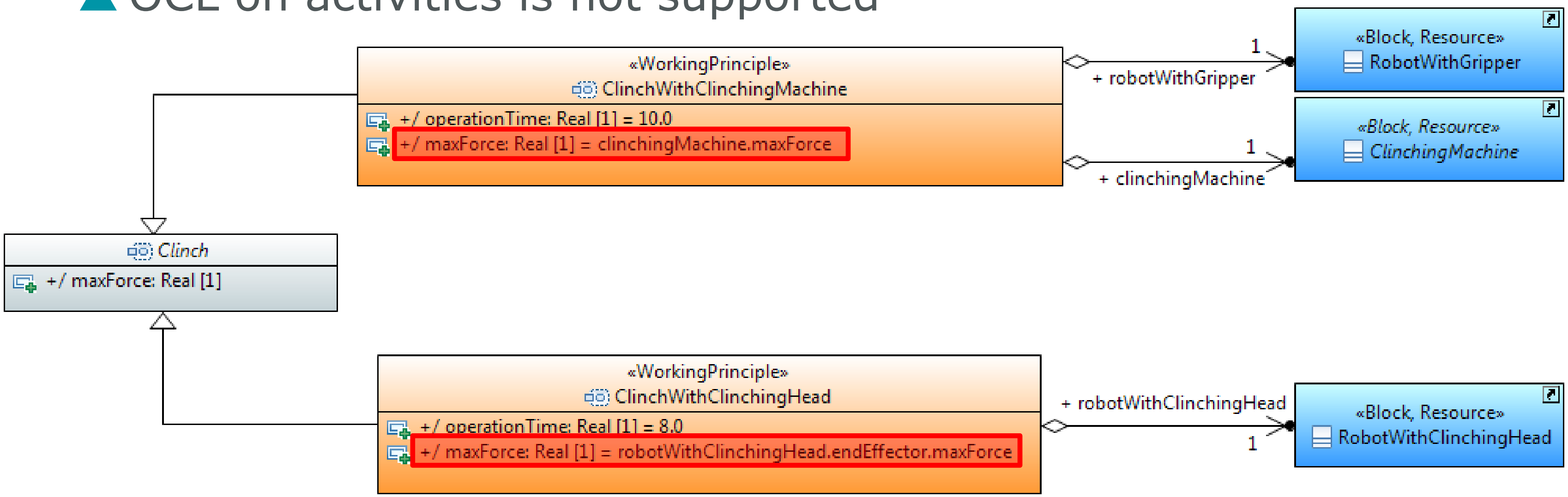
e.g. Backtransformed solutions





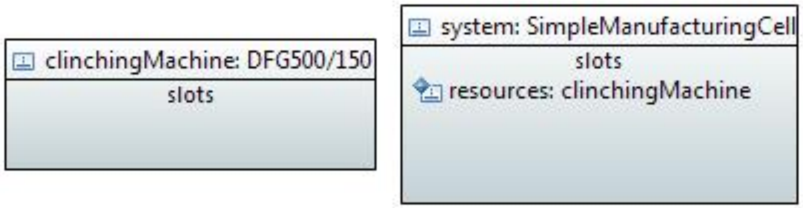
OCL framework

- OCL on activities is not supported



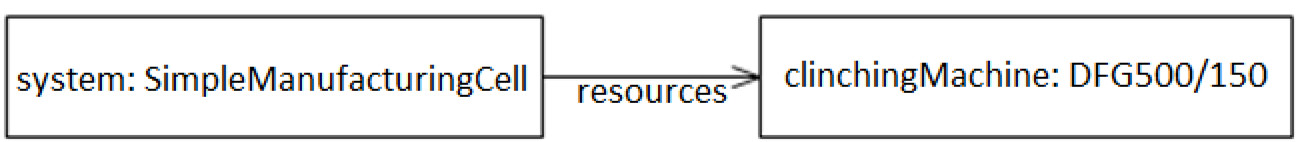
- Custom OCL translation
  - Manually created parser (monstrous spaghetti code)
  - A proper implementation would require an OCL metamodel

Instance creation and visualization



- Instances allow to
  - Specify partial solutions as 'expert knowledge'
  - Debugging through validation of correct and incorrect instances
  - Specification of context specific values

- Papyrus support for instances is lacking
  - Instance creation is tedious
    - Wizard, similar to MagicDraw, for creating feasible instances?
  - No proper visualization provided



Conclusions

- Papyrus can be used as part of a computational design space exploration toolchain
  - General purpose languages such as SysML+OCL contain most of the concepts necessary to express the design space
  - Usability & customizability is a **condicio sine qua non** in order to achieve adoption amongst industrial designers
  - Design space exploration tools still require interaction with the designer, which requires visualization of solution instances

- Papyrus seems to be a viable candidate to build upon, yet there remains a lot to do…

QUESTIONS?
REMARKS?

Thank you!